VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Metody umělé inteligence pro hru StarCraft

Methods of Artificial Intelligence for the Game StarCraft

2015 Jiří Hajný

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: Jiří Hajný

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Metody umělé inteligence pro hru StarCraft
Methods of Artificial Intelligence for the Game StarCraft

Zásady pro vypracování:

SSCAI je studentská soutěž metod umělé inteligence pro hru StarCraft pořádaná Univerzitou Komenského v Bratislavě. Hra StarCraft obsahuje tři rasy lišící se zvolenou strategií. Podle pravidel soutěže se metoda umělé inteligence musí zaměřit na jednu z těchto ras. Cílem bakalářské práce je navrhnout a implementovat metodu umělé inteligence pro rasu Prottos, která je specifická řízením velkého počtu slabších jednotek.

1. Proveďte rešerši metod umělé inteligence.
2. Seznamte se s pravidly hry StarCraft se zaměřením na danou rasu a popište strategie úspěšné pro boj s ostatními rasami.
3. Nastudujte rozhraní soutěže SSCAI.
4. Implementujte vybranou metodu umělé inteligence na základě nastudovaných poznatků o chování rasy Prottos.
5. Otestujte a porovnejte vaší implementaci s existujícími metodami přihlášenými do soutěže.

Seznam doporučené odborné literatury:

[1] RUSSELL, Stuart; NORVIG, Peter. AI a modern approach (3rd Edition). Learning, 2009.
[2] MILLINGTON, Ian; FUNGE, John. Artificial intelligence for games. CRC Press, 2009.
[3] Comenius University, Bratislava. Student StarCraft AI Tournament. 2014. url: http://www.sscaitournament.com/.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Roman Meca**

Datum zadání: 01.09.2014

Datum odevzdání: 07.05.2015

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě: 6. 5. 2015

Jiří Hajný

Rád bych na tomto místě poděkoval Ing. Romanovi Mecovi za jeho podporu a cenné rady při psaní této bakalářské práce.

Abstrakt

Tato bakalářská práce pojednává o Umělé Inteligenci ve real-time strategických hrách a její následné implementaci do hry StarCraft: Broodwar. Pro vytvoření Umělé Inteligence byly použity algoritmy Minmax, Hierarchical Tasks Networks a další. Na základě tohoto byla vytvořena samostatně fungující Umělá Inteligence s vlastním Micro a Macromanagementem. Tato práce by měla sloužit jako možný nástin pro všechny zájemce o tvorbu Umělé Inteligence ve hrách.

Klíčová slova: bakalářská práce, Umělá Inteligence, StarCraft: Broodwar, hry, Minmax, Hierarchical Tasks networks, real-time strategie

Abstract

This bachelor project deals about Artificial Inteligence in real-time strategy games and its implementation into the game StarCraft: Broodwar. To creation of Artificial Inteligence were used algorithms like Minmax, Hierarchical Task Networks etc. Bases on this there was created fully functional Artificial Inteligence with its own Micro and Macromanagement. This bachelor project can be used as potentional beginning place for those, who are interested in Artificial Inteligence in games.

Keywords: bachelor project, Artificial Inteligence, StarCraft: Broodwar, games, Minmax, Hierarchical Tasks networks, real-time strategy

Seznam použitých zkratek a symbolů

1vs1	–	1 versus 1 - typ bitvy, kdy proti sobě stojí dva oponenti
AI	–	angl. Artificial Inteligence(česky UI- Umělá Inteligence)
API	–	Application Programming Interface - rozhraní pro programování aplikací. Obsahuje sbírky procedur, které může programátor využívat.
BWAPI	–	API zprostředkující metody pro vytváření a upravování AI robotů pro hru StarCraft: Brood War
FPS	–	Frame Per Second(počet obrázků vykreslených na monitor za sekundu).
HP	–	angl. Hit Points (reprezentace života danné jednotky v čísle nebo procentech)
PSI	–	ve smyslu hry StarCraft psionická energie(nadpřirozená energie z vesmírů dovolující používat speciální schopnosti, či verbovat jednotky za rasu Protoss)
RTS	–	Real Time Strategy(Strategická hra odehrávající se v reálném čase).

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

Tato bakalářská práce pojednává o Umělé Inteligenci a její následné implementace do video hry Starcraft. Umělá Inteligence je v dnešním světě takřka na každém rohu. Ať už se jedná o domácnost, kde ji můžeme najít v myčkách na nádobí, pračkách, vysavačích či inteligentních sekačkách. Umělá inteligence se také nachází v dopravě, kdy vůz je řízen pouze počítačově (Google apod.), pomáhá při sestavování profilu pachatele trestného činu, či je nápomocná při studiu studentům. Složitější Umělá Inteligence se také nachází v robotice, kde v poslední době zaznamenala velkého pokroku (inteligentní roboti, sestava dronů apod.). V neposlední řadě se Umělá Inteligence aplikuje na video hry, kde je také pro její vývoj a testování ideální prostředí.

V předchozím odstavci jsem napsal, kde se Umělá Inteligence neboli AI vyskytuje. Avšak co to vlastně je AI? AI je vědní obor zabývající se vytvářením strojů nebo programů, které řeší úkoly podobným způsobem jako lidé. Jelikož se tato práce zabývá AI ve hrách, pak je na snadě uvést příklad z tohoto odvětví. Představme si na chvíli, že hrajeme středověkou hru. V této hře ovládáme postavu kováře, který musí ukout několik mečů pro svého pána. Když ukujeme meče a odneseme je k pánovi, tak od něj dostaneme další úkol. Po zadání úkolu se pán zvedne a vyráží spát do svého lože, nebo vyráží na noční lov. Jakým způsobem se pán rozhodl jít na lov, nebo do lože, když jej neovládá žádný člověk? Ano přesně tak, jeho AI o tom na základě jeho zdravotního stavu, aktuálního rozpoložení a denní doby rozhodla. Dalším příkladem může být dělník, který sbírá minerály potřebné pro stavbu jednotek a budov. Když při sběru minerál dojde, jakým způsobem se tento dělník rozhodne, zůstane stát, nebo vyhledá jiné naleziště daných minerálů v oblasti?

Tato práce by měla osvětlit základní pojmy Umělé Inteligence spolu s jejich složitějšími prvky použitelnými ve strategických hrách jako takových.

2 RTS hry

RTS hry, na které je soustředěna tato práce, jsou jedním ze subžánru strategických her, dalším takovým jsou tahové strategie. U RTS her se jedná většinou o vojenskou či stavitelskou simulaci při níž hráč ovládá více jednotek a budov v reálném čase. Na počátku hry hráč dostane určitý počet jednotek a budov nutných pro další rozvoj a je na něm, aby za co nejkratší dobu natěžil dostatek materiálů, postavil budovy a armádu a vyrazil do útoku na nepřítele. Toto všechno se odehrává při konstatních 24 FPS(Starcraft), kde hráč musí přidělit jednotkám co nejvíce příkazů, které za tuto dobu stihne. Hra se odehrává na generované mapě, na níž jsou většinou aplikované zákonitosti z reálného světa jako krytí za překážkami (StarCraft), zpomalení jednotek při průchodu různým terénem jako je voda(Original War), neschopnost vyjet příkré svahy pro lehké tanky(Original War), větší dohled a přesnější střelba pro jednotky stojící na vyvýšeném místě oproti jednotkám stojícím nížě(Starcraft, Original War, Rome 2 Total War) apod. Díky těmto zákonitostem se hra stává variabilnější a poskytuje mnoho možností jak vyhrát s menším počtem jednotek, které by bez těchto zákonitostí neměly šanci na přežití. Každá mapa také obsahuje místa strategického významu, kde se vyskytují suroviny pro těžbu(StarCraft, Stronghold), choke pointy- tedy zůžená místa, mezi oblastmi na mapě a další. Viditelnost na mapě je většinou limitována, kdy neprobádané oblasti jsou pokryty nepropustnou mlhou(StarCraft, Original War..), přes kterou nelze vidět ani terén. Jakmile jsou jednou tyto oblasti prozkoumány libovolnou jednotkou, tak se terén zviditelní, avšak pokud jednotka oblast opustí, pak nelze vidět to, co se na daném místě děje. Každá jednotka má okolo sebe rádius dohledu, v kterém vidí to co se děje aktuálně na daném místě. Následující výčet RTS her ukazuje jejich hlavní rysy

- DUNE II - hra je považována jako průkopník RTS her, kde hráč v reálném čase může najednou ovládat pouze jednu jednotku. Hra se dočkala přepracování v podobě DUNE 2000.
- StarCraft - ve hře se objevují neviditelné jednotky, které lze odhalit pouze jednotkami s scanovacími funkcemi. Hra také obsahuje speciální jednotky - Hrdiny. Hrdinové mají lepší schopnosti, než klasické jednotky, jsou také použiti pro prohloubení herního zážitku přidáním sekvencí videí podporujících děj.
- Warcraft 3 - hrdinové mají speciální schopnosti jako léčení či požehnání pro zvýšení obrany a útoku
- Original War - česká RTS. Ve hře se vyskytují pouze jednotky - Hrdinové, kdy každý hrdina má generované vlastnosti, které si průběhem hraní vylepšuje a tak se stává silnějším v různých odvětvích jako vojenství, mechanika, výzkum či stavebnictví. Pokud hrdina zemře, pak je nenávratně ztracen. Všem hrdinům dominuje hlavní postava, při jejíž smrti hra končí.
- Rome 2 Total War - Strategická hra kombinující vlastnosti tahové startegie pro globální mapu, kde hráč reprezentující starověký stát má na starosti obchodní a státní záležitosti mezi ním a sousedními státy a RTS pro bojové zóny, kde hráč v reálném

Obrázek 1: Ukázka ze hry Rome 2: Total war

čase ovládá skupiny svých jednotek a generála proti nepřátelské armádě. Všem jednotkám ve hře jsou přiděleny specifické proměnné vlastnosti měnící se v závislosti na průběhu bitvy jako je vyčerpání, nebo morálka stoupající nebo klesající podle úrovně jednotek, poměru jednotek mezi nepřátelskými jednotkami a vlastními, vyčerpáním či smrti jednoho z generálů.

3 Hra StarCraft: Brood War

Hra StarCraft: Brood War je Real Time Strategy (neboli RTS) rozšíření základní hry StarCraft. Byla vytvořena společností Blizzard Entertainment ve spolupráci s Saffire Entertainment. Ve hře spolu soupeří tři rasy **Protoss**, **Terran** a **Zerg** o celkovou nadvládu nad mapou.

3.0.1 Herní mechanismus

Jakožto RTS hraje hráč úlohu vrchního velitele dané rasy, za kterou aktuálně hraje. Staví základny, verbuje armádu a vylepšuje jednotky. Omezením pro hráče v dané situaci je dostatek minerálů (slouží pro stavbu většiny budov a jednotek), plynu (je potřeba pro vylepšení jednotek, stavbu některých budov a jednotek) a supply bodů (dá se prezentovat jako dostatek místa pro ubytování armady, každá rasa má jiné pojmenování).

Herní mapa je rozdělena na menší lokality od sebe odděleny úzkými přechody (choke pointy), ve kterých se všechny postupující jednotky zpomalí (za určitý čas je možné jen pro několik jednotek projít chokepointem). Mapa dále využívá přírodních překážek jako jsou hory nebo moře pro oddělení jednotlivých hráčů a vyvýšená území, z nichž střílející jednotky útočící na jednotky položené níže mají větší šanci na zásah.

3.0.2 Rasa: Terran

Rasa Terran je lidská rasa s průměrnými jednotkami. To z ní činí dobře adaptovatelnou rasu na jakoukoliv strategii. Má vylepšenou obranu budov, obrané bunkry a silné tanky, které mají dva módy pro aktuální situaci.

Výhody rasy Terran:

- průměrná cena jednotek
- budovy mohou být postaveny kdekoliv
- mnoho budov může vzlétnout a přesunout se
- rasa má silnou obranu (věže, bunkry pro vojáky nebo tanky v siege módu)
- budovy a jednotky se mohou léčit/ opravovat
- rasa Terran disponuje nejničivější zbraní - nukleární bombou

Nevýhody rasy Terran:

- průměrná cena jednotek
- nukleární bomba vyžaduje velké množství výzkumu a surovin

Z předchozího výčtu lze vyčíst, že rasa Terran je nejvíce v Mid Game nebo Late Game nebezpečná, když začíná vyrábět těžké obléhací tanky a stíhačky. [9]

Obrázek 2: Ukázka prostředí hry StarCraft: Broodwar

3.0.3 Rasa: Zerg

Rasa Zerg je mimozemská rasa nehumanoidního charakteru. Její jednotky jsou levné na výrobu a mají malé rychlé jednotky s nevalnou útočnou silou, proto se při hraní této rasy spoléhá hráč především na kvantitu svých jednotek.

Výhody rasy Zerg:

- velmi levné jednotky
- flexibilní produkce jednotek (jsou vyráběny pouze v jedné budově)
- rychlejší expanze základny

Nevýhody rasy Zerg:

- budovy musí být vyrobeny na creep jednotkách
- Dron(dělník pro rasu Zerg) je při výrobě budovy obětován

Rasa Zerg útočí především v množství při nejnižší viditelnosti (dokáží se zahrabat pod zem). [9]

3.0.4 Rasa: Protoss

Rasa Protoss je starodávná humanoidní rasa(rasa mající společné prvky s lidskou rasou) pocházející z planety Aiur. Protossané mají velmi vyspělé technologie (regenerativní štíty, útok) a dokáží využívat PSI energie k sesílání kouzel. Rasa Protoss má jedny z nejsilnějších základních jednotek Protoss Zealot, které jsou v počáteční fázi hry velice účinní

především díky svým automaticky se doplňujícím štítům a silnému útoku na blízko.

Výhody rasy Protoss:

- silné jednotky
- Probe(dělník) pouze zahájí stavbu budovy, ta se dokončí sama
- všechny jednotky a budovy jsou vybaveny regenerativním štítem
- možnost vyrobit si plně neviditelné jednotky

Nevýhody rasy Protoss:

- jednotky ani budovy nemohou být léčeny
- dražší jednotky
- budovy musí být zbudovány v blízkosti Pylonu, pokud je Pylon zničen, pak budova ztrácí schopnost produkce nebo střelby (jedná li se o věž)

3.1 Strategie rasy Protoss

Ve StarCraftu je Protoss rasa považována za Early Game nebo Mid Game rasu tzn. nejvíce nebezpečná bývá ze začátku, kdy jednotky protos mají silné pancéřování a vysoký útok v porovnání s ostatními rasami. Její citlivá místa jsou především v samotných Pylonech (budova sloužící pro navýšení kapacity jednotek a stavbu budov v Pylon blízkosti), které pokud jsou zničeny, tak může nastat ochromení výroby nových budov a jednotek. [9] Z tohoto důvodu bylo vymyšleno několik základních strategií pro boj proti jednotlivým rasám.

3.1.0.1 Protoss versus Zerg Při souboji Protoss versus Zerg je možno použít široké spektrum strategií díky různorodým jednotkám a budovám, které zabezpečují obranu základny. Protoss jednotky jsou oproti Zerg jednotkám pomalejší, silnější, mají více HP a déle se trénují. Ideálním případem tedy je, kdy Protoss armáda se skládá z jednotek na blízko i na dálku s pokud možno co největším rozptylem síly. V klasických hrách lidských protivníků 1vs1 si oba hráčí přisvojí strategii nepřítele a poté se dle ní do budoucna řídí. [9]

3.1.0.2 Protoss versus Terran Protoss versus Terran je jedna z nejstatičtějších typů her, které se v 1vs1 bitvách vyskytují. Protoss hráč by se měl ihned ze začátku snažit o ovládnutí prostoru okolo Terran hráče tak, aby Terran nemohl obsadit další minerály. V počátečních fázích hry má Protoss hráč rychlejší a lépe vybavené jednotky, kdežto v pozdějších fázích hry má navrch ve všech směrech Terran. Proto je potřeba, aby Protoss hráč zvítězil v co nejkretší době, kdy je to ještě možné. [9]

3.1.0.3 Protoss versus Protoss

Protoss versus Protoss je jeden z nejpomalejších typů 1vs1 bitev. Díky stejným rasám zde nejsou jednotky, které by měly navrch nad jinýma a proto je zde větší volnost při rozhodování které jednotky postavit a jak vést útok. Tak jako proti Terran se Protoss hráč snaží již od počátku o dominanci a blokádu svého protějšku, kdy v pozdějších fázích hry jej pokoří kvantitou. [9]

3.2 Zvolená strategie za rasu Protoss

Z předcházejících poznámek o boji rasy Protoss proti ostatním rasám lze vyčíst jedna základní věc. Rasa Protoss je bezesporu early game - rush rasa, neboli rasa mající největší převahu a tedy šanci na výhru v počátečních fázích hry a to především díky velmi odolným jednotkám Protoss Zealot a Protoss Dragoon, které mají velmi silné štíty. Proto byla snaha zavést jednotný způsob boje lišící se pouze rozpoložením a stavbou budov.

Při změně ras, proti kterým AI bojuje záleží především na build orderu budov, tedy posloupnosti stavby budov. Při pozorování soubojů hráčů proti sobě byl pro každou rasu sestaven build order napomáhající větší úspěšnosti ve hře. Ku příkladu rasa Zerg využívá nejčastěji neviditelné jednotky napadající nepřátelské základny. Nejůčinější obrana proti tomuto je postavit větší množství obraných věží, které jsou schopny detekovat a zničit neviditelné jednotky ještě před vpádem do vlastní základny.

Při stavbě bojových jednotek se spoléhá na znalost aktuálního nepřítele a jeho nejčastějších jednotek. Proti těmto jednotkám se vytvoří tzn. counter jednotky (jednotky mající největší úspěšnost při boji s touto jednotkou). Nejčastější budovanou jednotkou však je Protoss Dragon, který díky svým zbraňovým systémům útočícím na dálku na letecké i pozemní cíle a okamžitém otočení do potřebném směru chůze se stává ideální bojovou jednotkou.

Při boji samotném se používá technika zvaná **harassment**, tedy obtěžování, z které těží převážně jednotky bojující palnýma zbraněma na dálku (Protoss Dragoon). Technika harassment spočívá v zasazení rány nepříteli a okamžitém ústupu. Mezi největší **výhody** této techniky se řadí:

- Při větším počtu jednotek používající harassment dochází k efektu taktickému ústupu, kdy ustupující jednotka je krytá palbou druhou jednotkou. Tento efekt má za následek větší ztráty nepřítele(převážně pokud se jedná o nepřítele bojujícího na blízko).
- Při použití jednotek Protoss Dragoon je zvýšená účinnost boje především díky faktu, že tyto jednotky se nemusí otáčet pro možnost ústupu (ostatní jednotky se vždy musí otočit na míste do požadovaného směru a až poté mohou nabrat plnou rychlost při které se na ústupové místo dostanou. Tímto faktem se jednotky nacházejí delší dobu v místě střetu a zvyšují tak pravděpodobnost zásahu a zničení).
- Při klasické bojové strategii útočí většínou všechny jednotky na nejbližšího nepřítele, aby byl co nejdříve zničen. Pokud ale jednotky ustupují, pak se nejbližší cíl jednotek mění, takže nedochází k velkým ztrátám na jednotkách.

Mezi nejvetší **nevýhody** harassmentu se řadí:

- Při velkém počtu jednotek na jednom místě nebo v choke pointu dochází k zablokování ústupujících jednotek jednotkami útočícími. Tímto může docházet k zbytečně velkým ztrátám.
- Při útoku jednotek používajících harassment na jednotky stojící na vyvýšeném místě dochází k velké ztrátovosti, jelikož pravdepodobnost jednotky útočící z nížší polohy na jednotky postavenou výše je pouze 53.125% [12].

I přes nevýhody této techniky se jedná o techniku, která se s úspěchem dá aplikovat na většinu konfliktů u všech ras ve hře.

4 Umělá Inteligence

4.1 Co je to Umělá Inteligence?

Základní AI se rozděluje na 8 částí viz. Tabulka 1 . Definice v horní části tabulky se zaměřují na zpracování a uvažování, zatímco spodní část se zaměřuje na chování. Definice v levo se zaměřují více na lidské pohnutky, zatímco definice vpravo na racionální chování [5].

4.1.1 Agenti

Agentem se stává jakákoliv samostatná jednotka AI, která pomocí sensorů dokáže vnímat prostředí, ve kterém se vyskytuje a na základě toho jednat. **Sensorem** se stává jakákoliv součást agenta, která mu dá na vědomí informace z vnějšího prostředí. V reálném světě se může jednat o teploměry umístěné na povrchu robota, tlakové snímače, sonar či termovizi. V počítačovém světe se může jednat o události, které jsou volané v určitém okamžiku např. při ukončení naplňování zásobníku čísel do pole pro agenta počítajícího určité operace, nebo oznámení o vytvoření či zničení jednotky ve počítačové hře. Díky sensorům tedy agent může adekvátně reagovat na aktuální situaci v daném prostředí. Pokud agentovy sensory dávají plný přístup k prostředí, pak je prostředí **plně pozorovatelné**, v jiném případě pouze **částečne pozorovatelné**.

Prostředí může být **statické** tj. neměnící se nebo **dynamické**, které se neustále mění. Dále se prostředí dělí na **deterministické** tedy plně pozorovatelné a **stochastické** tedy z části pozorovatelné.

Každý agent také obsahuje určitou znalost prostředí v němž se vyskytuje. **Znalost prostředí** jsou vědomosti agenta, které se aplikují na získané informace ze sensorů. Pokud agent robot pracující poblíž lávového pole zjistí, že teplota okolí se blíží teplotě tání materiálu, z něhož je agent vyroben, pak pomocí této vedomosti se může agent rozhodnout pro následující akce. V předešlé věte bylo zmíněno, že agent se může rozhodnout na základě vědomostí pro následující akci, ale zároveň nemusí udělat nic. Toto na první pohled zvláštní chování je totiž podmíněno agentovymi cíly. Každý agent musí mít hlavní **cíl**, tedy něco, pro co byl stvořen. Agentovo chování je podmíněno cílem, kterého se snaží dosáhnout. V reálném světě se může jednat o robota, který má za úkol přecházet silnici, nebo herního agenta jehož cílem je vyhrát hru.

4.2 Umělá inteligence ve strategických hrách

RTS strategie se řadí do kategorie **her dvou hráčů, zero-sum**, kde na konci vždy jeden hráč vyhraje a druhý prohraje, nebo kde součet odměn hráčů je roven nule. Two Player Zero Sum hry obsahují tyto vlastnosti:

- Dva hráče- vlastní hráč A(Max) a nepřátelský hráč B(Min).
- Sadu stavů ve hře.
- Počáteřní stav, kde hra začíná

Myslet lidsky	**Myslet racionálně**
"Vzrušující je snaha nechat počítače myslet..**stroje s mozkem** v plném znění "(Haugeland, 1985) "[Automatizace] aktivit, které asociujeme s lidským myšlením jako rozhodování se, nalézt problém, učit se.."(Hellman, 1978)	"Studium duševních schopností použitím výpočetních modelů"(Charniak and McDermott, 1985) "Studium výpočtů, které umožnují vnímat, přemýšlet a jednat"(Winston, 1992)
Chovat se lidsky	**Chovat se racionálně**
"Umění vytvářet stroje, které provádějíí funkce takove, jež potřebují určitou inteligenci lidí"(Kurzweil, 1990) "Studium jak udělat počítače dělat věci, ve kterých jsou lidé lepši"(Rich and Knight, 1991)	"Výpočetní inteligence je studium návrhu inteligentních agentů"(Poole, 1998) "AI se zabývá inteligentním chováním artefaktů"(Nilsson, 1998)

Tabulka 1: Základní typy AI

- Ukončovací stav určící výhru jednoho hráče a porážku druhého hráče.
- Sadu kroků vedoucích od hráče A k hráči B a naopak (jedná se o orientovaný strom)
- Utility funkci značící, zda-li se krok vyplatí.

Umělá inteligence v RTS se dělí na dva hlavní proudy. **Strategické rozhodování** 4.4 a **Taktické rozhodování** 4.3. Tyto dva proudy mohou být vyřešeny libovolnými algoritmy, kde na výstupu by spolu měly nějakým způsobem komunikovat.

4.3 Taktické rozhodování

Taktické rozhodování neboli micromanagament zahrnuje ovládání všech bojových jednotek v akci. Jedná se o nejdůležitější součást AI ve strategických hrách, jelikož rozpoložení a styl útoku může zvýhodnit i nejslabší národ. V následujících podkapitolách jsou rozepsány nejčastější typy zpracování AI vyskytující se v RTS video hrách.

4.3.1 Reinforcement learning

Reinforcement learning problem volně přeloženo pokus-omyl. Jedná se o Agenta, který se učí ze svých hlavních chyb v plně pozorovatelném prostředí na základě odměn. Na počátku má agent k dyspozici sadu možných stavů a možných akcí, které může provést.

Po každé agentově akci je prozkoumán stav prostředí, ve kterém se nachází a agent dostane odměnu. Odměna může být neutrální, záporná, nebo kladná. Tímto se agent naučí, zda-li v podobných situácích se vyplatí použít příslušné akce.[7]
Nevýhody Reinforcement learning

- Vyskytuje se zde neblahý **problém přisuzování viny**. Jedná se o problém, kdy agent neví, která akce byla zodpovědná za kladnou nebo zápornou odměnu, pokud se oběvila dlouhou dobu před aktuálním stavem.
- I když se nemusí jednat o dynamické prostředí, tak agentovy akce mohou nabýt efektu až v budoucnu, což v aktuální situaci agent nemusí poznat.
- Nastává **dilema prozkoumat-využít**. Pokud agent udělá za sebou několik dobrých akcí, pak by se měl rozhodnout, jestli bude pokračovat v aktuálních akcích, nebo nalézt lepší akce? Je možné, že agentovo chování může mít větší efektivitu o které agent neví.

Algoritmus byl použit programátorem Shantia, Begue a Wiering, kde algoritmus Sarsa byl pro učení se ovládat malé množství jednotek v boji. Algoritmus byl použit s využitím neurálních sítí pro zjištění nejlepší možné odměny při útoku nebo útěku jednotek. Tato technika avšak není využívána ve větší míře ve větších bitvách[6]. Reinforcement learning nepatří přímo do problémů týkajících se her, jelikož nebere přímo v potaz druhého hráče jako rovného protivníka. Avšak i přes to je možné jej aplikovat.

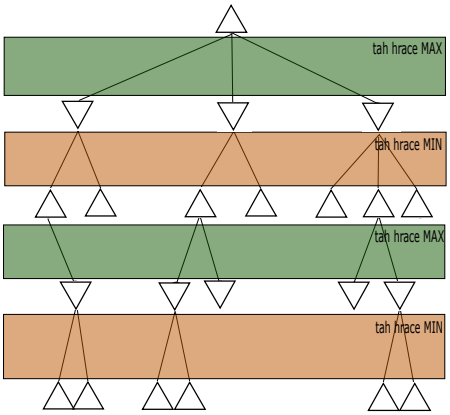
4.3.2 Minmax-Tree

Mezi mnohé implementace AI se řadí Minmax strom a k němu náležícím minmax algoritmus. Díky tomuto stromu je možno implementovat určitou míru predikce v AI. Jakožto každý orientovaný strom se i minimax-tree skládá z uzlů spojených větvemi. Kořenový uzel představuje počáteční stav situace, kterou je za potřebí řešit a jeho větve jsou možné akce, které lze na danou situaci aplikovat. Tyto akce vyůsťují v další uzly - stavy nadcházející po provedení akce předchozího uzlu. Základní vlastnost Minmax stromu je jeho přizpůsobení pro hru dvou hráčů, kde každá hrana určuje zda-li táhnul hráč (Max), nebo nepřítel(Min)[4] viz. obrázek 3.

Mezi hlavní nevýhodu Minmax stromu patří fakt, že je originálně navržen pro plně pozorovatelné a informativní hry jako šachy, dáma apod.. V těchto hrách neexistuje ohledně herního prostředí a možných akcí žádná neznámá, což se bohužel nedá říci u RTS video her.

4.3.3 Minmax algoritmus

Základní algoritmus pro řešení hry dvou hráčů MAX a MIN. Pro práci používá Minmax strom, v němž každý uzel představuje výsledek tahu hráče MAX nebo MIN ve **hře**. Jak již bylo popsáno 4.2 skládá se z počátečního stavu, množiny možných kroků, které hráč může použít, ukončovacího stavu a utility funkce, jež spočítá heuristickou hodnotu pro hráče. Narozdíl od klasického problému, kde figuruje pouze jeden agent, který vyhledá



Obrázek 3: Minmax strom. Trojůhelník špičkou nahoru představuje tah hráče MAX, špička dolů hráče MIN

ve stromu nejlepší řešení tento problém zahrnuje také protivníka druhého hráče nebo agenta MIN. Tento protivník má do vyhledávání také co říci a proto MAX hráč musí vyhledat takovou strategii, která zahrnujíc protivníka MIN dovede hráče MAX k vítězství, nebo k nejlepšímu možnému stavu. Minimax algoritmus se skládá z následujících kroků[4]:

- Vygenerování herního/minmax stromu.
- Aplikací utilit funkce na všechny okrajové listy, čímž se zjistí finální stav po všech krocích vedoucích k němu. U RTS video her z důvodu složitosti a množství jednotek a budov se jedná o výpočet heuristické hodnoty konečných stavů. Utilit hodnota přiřadí každému konečnému uzlu určitou číselnou hodnotu, kde čím větší číslo, tím lepší.
- Od spodu porovnávání jednotlivých hodnot v závislosti na tom, zda-li je na tahu hráč MAX nebo MIN. Hráč MIN požaduje nejnižší hodnoty, proto pokud se jedná o jeho uzel, pak se porovnají hodnoty a vybere se nejnižší. Hráč MAX provádí stejné provonávání s odlišením toho, že chce největší hodnoty.

Výhodou Minmax algoritmu je ta, že hráč MAX nalezne vždy optimální cestu, která vytěží z dané situace maximum. Nevýhodou je především doba zpracování a generování, kdy průchod algoritmem (Minmax algoritmus prochází všechny uzly ve stromu pro nalezení optimální strategie) má časovou složitost O(n), kde n je počet uzlů ve stromu[4].

4.3.4 Alpha Beta prořezávání

Negativní vlastnost algoritmu Minmax prohledávání všech uzlů a tím navyšování času zpracování je možno zkrátit až o polovinu použitím algoritmu Alpha-Beta Prořezávání.

Tento algoritmus je téměř totožný s algoritmem Minmax, avšak počítá nově s dvouma proměnnýma Alpha a Beta. Idea prořezávání spočívá ve faktu, že hráč MIN zvolí vždy cestu s nejmenším možným přínosem pro hráče MAX[4]. Na příkladu boje jednotek, pokud akce jednotek hráče MAX povede k zničení několika jednotek od hráče MIN a odpověď MIN povede ke zničení všech hráčových jednotek, pak je určitě nasnadě nevést proti nepříteli akci, kterou si hráč MAX zničí všechny své jednotky, ale provést akci, při které hráč bude vědět, že přijde o méně svých jednotek. Díky tomuto způsobu je možno se vyhnout počítání až půlky uzlů u kterých je jasné, že povedou k větším ztrátám, než ostatní.

Čas algoritmu Alpha-Beta prořezávání je možno ještě zrychlit vhodným seřazením při tvorbě stromu, toto avšak není u RTS video her zcela ideální především z důvodu výpočtů v reálném čase, kdy vhodné řazení uzlů zabere další čas.

4.3.5 Monte Carlo Planning

Při aplikaci Minmax stromu, určeného zejména pro otevřenou hru dvou hráčů jako dáma nebo šachy, v RTS video hrách kde není plně pozorovatelné herní prostředí a akce mohou generovat nespočetné množství stavů dochází k chybám, které mohou vyůstit v rychlou porážku nepřítelem, který ani nemusí používat AI na vysoké úrovni[3].

Výhodou Monte Carlo Planning jsou minimální požadavky na analýzu a výpočty dané situace. Skládá se ze dvou kroků:

- Vytvoření možných akcí pro hráče pomocí **stochastického samplování**(vytvoření plánu i za pomocí ne plně popsaného a pozorovatelného prostředí).
- Výběru plánu, jež má statisticky největší možný přínos pro hráče.

Monte Carlo Planning počítá pouze s několika základními typy akcí jako útok, ústup, postup k cíli apod., z těchto akcí za pomocí samplování získá evaluační hodnotu. Vyhledávání poté může samplovat akce s různými parametry na vstupu pro všechny hráče. Nakonec zvolí akci, která má pro hráče nejlepší možné výsledky. Jakožto u všech AI algoritmů je i v Monte Carlo Planning zapotřebí určitá míra abstrakce závisející na pomeru objemu dat zpracovaných za určitý čas.[3]. Monte Carlo Planning byl také s úspěchem použit ve hře Rome 2: Total War.

4.4 Strategické rozhodování

Strategické rozhodování neboli macromanagement v sobě skrývá schopnost operovat a maximalizovat zisk z ekonomické stránky hry. Tohoto však nelze vždy úspěšně dosáhnout především díky mlze na mapě, která může zakrývat nepřátelské jednotky, s kterými tedy nelze počítat. Pro zjednodušení výpočtů jsou do her přidávány plánovací systémy, které podobně jako hráč čekají na nějakou událost(časový interval od počátku hry, dostavení budovy, získání určitého počtu materiálů).

4.4.1 Case Based Planning

Jedná se o techniku plánovaní akcí pro aktuální situaci s pomocí využití hledání podobných situací nastalých v minulosti. Na základě techto situací je možno vyvarovat se nekterých minulých chyb a díky tomu zlepšit své chování také do budoucna [6]. Tento druh plánování uchovává poznatky z dřívějších situací především v log souborech, kde později vyhledá podobnou situaci. Někdy se pro zrychlení vyhledávání používají vyhledávací stromy.

4.4.2 Hierarchical Tasks Networks

Dalším způsobem jak je možné řešit problém je Hierarchické plánování. Toto plánování spočívá v hierarchické rozdělení problému, kdy plánovací systém řeší každou jednotlivou jeho část při použití určité vrstvy abstrakce. Rozdělení problému je výhodné kvůli zjednodušení, avšak složitejší při řízení řešení mezi různými abstrakčními vrstvami. Tento problém komunikace řeší Hierarchical Tasks Networks (HTN), což jsou sítě obsahující úkoly, jejich řazení a metody sloužící k jejich dosažení [6].

HTN plánování může být použito pro hledání primitivních řešení, kde se především zaměřuje na implementaci akce, která má za úkol dosažení nějakého cíle. Výhoda HTN při hledání primitivních řešení je především snadná srozumitelnost pro člověka [4].

Jak již bylo napsáno o pár řádků výše, Hierarchické plánování je možné rozdělit do více vrstev, nejspodnější vrstva se zabývá hledáním primitivních řešení. Je tedy na snadě přidat do řešení další, více abstraktní vrstvu, která se již bude zabývat pouze globálními problémy např.: Agent má za úkol dojet na letiště bez popisu cesty. Tato vrstva již neřeší přímo akce ve smyslu implementace, ale pojímá celý problém globálně [4].

4.4.3 Goal-Driven Autonomy

Jedním z dalších způsobů implmentace agenta je Goal-Driven Autonomy(GDA). Jedná se o agenta, který je schopen přemýšlet jak o světě, ve kterém se vyskytuje, tak o sobě samém. Takový agent se může skládat z následujících částí [8]:

- **Detektor rozporu** generující rozpor pokaždé, když je očekávaný zisk z nějaké akce změněn k horšímu. Ve hře může nastat, když nepřítel staví nový typ jednotek, které mají jinou strategii, nebo nepřítel expanduje ve stavění svých základen.
- **Generátor objasnění** generující na základě detektoru rozporu a svých vnitřních znalostí událost, která objasní o jakou změnu se jedná.
- **Formulátor cíle** tvořící na základě objasnění nové cíle pro agenta jako jsou vykonání nové strategie boje, stavby budov, zaútočení všech jednotek na danou novou jednotku či budovu apod.
- **Manažer cílů** vybírající nové cíle k uskutečnení na základě nových předchozích poznatků.
- Plánovač volící pořadí nových cílů.

5 Implementace Umělé Inteligence

5.1 BWAPI

BWAPI (Brood War API) je API používající se pro programování AI do hry StarCraft Brood War. Výstupem z BWAPI je knihovna AIModule.dll, která se nahrává do složky se StarCraft hrou, jež při spuštění nahraje všechny dll moduly s AI a přidá je do hry jakožto další hráče. BWAPI obsahuje několik tříd pro práci s hrou samotnou či ovládáním jednotek. Výčet nejdůležitějších tříd BWAPI:

- **AIModule** - třída obsahující nejdůležitější Eventy ve hře jako OnStart(), OnEnd(), OnUnitCreate() apod.
- **Game** - abstraktní třída sloužící pro získávání informací aktuální probíhající Broodwar hry zahrnující informace o hráči, jednotkách či mapě.
- **Player** - třída poskytující informace o hráči/AI obsahující metody pro získání informací o vytěžených materiálech či informace o vylepšeních.
- **Unit** - třída používající se pro získání informací o individuální jednotce v aktuální hře a rozkazech, které mohou být uděleny aktuální jednotce jako zjištění typu jednotky, přidělení rozkazu útoku apod.
- **UnitType** - třída pro získání informací o typu dané jednotky(Unit). Obsahuje metody o zjištění ceny jednotky, její počáteční hodnoty HP, maximální rychlosti, výpisu zbraní jednotky apod.
- **WeaponType** - třída poskytující informace o zbraňovém vybavení obsahující metody pro zjištění cooldownu zbraně, jejího útoku, vzdálenosti potřebné pro střelbu apod.

Aktuálně existuje nejnovější verze BWAPI 4.0.1 BETA, nicméně pro účel této práce byla zvolena starší stabilnější verze 3.7.4.[1]

5.1.1 Obsah CD

Přiložené CD obsahuje samotné BWAPI uložené v kořenové složce BWAPI. Dále se na kořenovém adresáři nachází složka Code obsahující hlavičkové soubory implementující samotného AI robota a dll knihovnu ExampleAIModule.dll.

Obsah složky Code:

- **AbstractedTeams.h** obsahuje třídu AbstractedTeams zodpovědnou za abstraktní interpretaci všech jednotek.
- **Building.h** obsahuje třídu Building, která zahrnuje práci s vlastními budovami, jejich build order apod.

[1] Z důvodů přechodu na nový systém online dokumentace je možné, že reference nebudou fungovat.

- **ExampleAIModule.h** a **ExampleAIModule.cpp**. Tyto dva soubory již byly implementovány v BWAPI. Slouží jako základní třída, která je volána ihned po startu hry StarCraft a obsahují implementaci událostí volaných na robotovi. V této třídě se vytváří instance třídy SMap, Micromanagement, Macromanagement a Scout, jejichž metody Loop se volají každý probíhající cyklus hry.
- **GameTree.h** a v ní implementovaná třída GameTree vytváří Minmax strom abstraktních jednotek použitý pro predikci boje. Nachází se zde také metoda **alphaBeta** sloužíci pro průchod stromu a zvolení nejlepší strategie.
- **Macromanagement.h** obsahuje třídu Macromanagement zodpovědnou za celou ekonomickou část hry. Tato třída vytváří nové jednotky, staví budovy a posílá dělníky težit suroviny.
- **Micromanagement.h** se třídou Micromanagement zodpovědnou za bojovou stránku hry. Třída za pomocí predikce herního stromu určuje následující akce všech jednotek.
- **MyFlyUnit.h** a **MyGroundUnit.h**, které shlukují bojové jednotky daných typů (letecké a pozemní).
- **Scout.h** a v ní třídu Scout zodpovědnou za ovládání průzkumných jednotek.
- **SMap.h** se stejnojmenou třídou. Tato třída je základní třída obsahující většinu podružených tříd jako MyGroundUnit, Building, Worker a je užívána nadřazenými třídami jako Scout, Micromanagement a Macromanagement. Dále jsou v ní implementovány metody na reakci na události volané v třídě ExampleAIModule.
- **State.h** a v ní třídu State vyjadřující jednotlivé stavy stromu Minmax- útok a ústup.
- **Worker.h** a třídu Worker implementující ovládání dělníků Protoss Probe.

5.2 Macromanagement

Macromanagement má na starosti celou ekonomickou část inteligence. Výpočetní části managementu se provádějí pouze pokud mineral suroviny hráče narostou na hodnotu větší, než 50. Každá část je závislá na aktuálních surovinách, popř. dostupnosti dělníků(stavba budov). Pokud je tedy potřeba zbudovat budova vyžadující dělníka a 200 jednotek minerálů, pak celý proces stavby je pozastaven do doby, než se nenatěží potřebné množství surovin a neuvolní se volný dělník. Třída Macromanagement má za úkol následující operace:

- verbování dělníků
- stavbu potřebných budov
- stavbu plánovaných budov

- stavbu jednotek/vylepšení

Pro řešení problému macromanagementu bylo použito hierarchické plánování, které rozdělilo každé dílčí operace na menší podoperace řešící vždy část problému. Všechny tyto operace jsou vykonávany podle důležitosti ve funkci **BuildLoop()** viz. kód 1 za sebou řazeny ve smyčce ve třídě **Macromanagement**. Jelikož se jedná o strategickou hru, tak je velmi důležité mít dělníky, kteří budou produkovat suroviny důležité pro další postup hrou(metoda starající se o verbování dělníků - **BuildUnit(BWAPI::UnitType unitType)** 2). Ihned po dělnících je nejdůležitější hlavní budova pro rasu Protoss - **Protoss Nexus**, která je hlavní skladiště pro suroviny a také slouží pro verbování dělníků.
Dále se třída Macromanagement stará o rozdělení práce dělníkům.

```
void BuildLoop()
{
    BWAPI::UnitType worker = BWAPI::UnitTypes::getUnitType("Protoss_Probe");
    BuildUnit(worker);
    if (!BuildUsefullBuildings())
    {return;}

    int nexusOutput = BuildNexus();
    int MinusMinerals = 0;
    if (nexusOutput == 1)
    {}
    else if (nexusOutput == 0)
    {return;}
    else
    {minusMinerals = nexusOutput;}

    BuildBuilgind();

    BWAPI::UnitTYpe nextUnit = ChooseUnit();
    BWAPI::UnitType nextUpgrade = ChooseUpgrade();

    if (UnitIsBetterThenUpgrade(nextUnit, nextupgrade))
    {
        BuildUnit(nextUnit);
    }
    else
    {
        BuildUpgrade(nextUpgrade);
    }
}
```

Výpis 1: Metoda BuildLoop()

5.2.1 Verbování jednotek

Verbování všech jednotek probíhá ve funkci **BuildUnit(BWAPI::UnitType unitType)** viz kód 2. Vstupem do metody je typ jednotky, který chceme vyrobit. Jako první si metoda načte všechny budovy, které mohou vyrábět danou jednotku(nezahrnuje budovy, které

již nějakou jednotku vyrábějí). Poté, pokud je jejich počet větší než nula a je dostatečný počet minerálů na skladě se vydá příkaz k budování dané jednotky. Pokud se požadovaná jednotka jmenuje Protoss Probe(dělník za rasu Protoss), pak se zavolá metoda **IsUsefullToBuildWorker()** počítající vhodnost stavby dělníka(pokud výdaje hráče převyšují požadované příjmy, pak výstup je true).

```
void BuildUnit(BWAPI::UnitType unitType)
{
    std :: list <BWAPI::Unit*> idleBuildings = GetIdleBuildings(unitTYpe);

    if ( idleBuildings . size ()  == 0)
    {
        return;
    }

    if (unitType.getName() == PROTOSS_PROBE)
    {
        if (! IsUsefullToBuildWorker())
        {
            return;
        }
    }

    if (! IsAvalibleMinerals (unitType))
    {
        return;
    }

    OrderToBuildUnit(idleBuildings, unitType);
}
```

Výpis 2: Metoda **BuildUnit(BWAPI::UnitType unitType)** sloužící pro výrobu jednotek

5.2.2 Stavba potřebných budov

Stavba budov za rasu Protoss se dá rozdělit do dvou částí:

- stavba budov dle rozkazu
- stavba potřebných budov

Stavba potřebných budov se zabývá stavbou budov, jejich potřeba je dána dynamickým rozvojem hráčovy základny či stavem nepřítele a zapadají do ní budovy **Protoss Nexus**, **Protoss Pylon**, **Protoss Assimilator** a **Protoss Photon Canon**. Stavbou těchto budov se zabývá metoda **BuildUsefullBuildings()** viz. kód 3. Každá budova má specifické podmínky pro stavbu, proto každá má své vlastní bool metody, které rozhodují, jestli se má daná budova postavit. Protoss Nexus se staví pouze v případě nedostatku surovin v blízkosti základny. Pylon je potřeba pro stavbu jednotek a budov, proto se staví v případě, že je nedostatek supply bodů(nemohu postavit další jednotky). Každá základna také potřebuje svůj vlastní Assimilátor (zařízení pro těžbu plynu). Obrana základny je

nutná v případě, kdy nepřítel má neviditelné jednotky, což se může stát, proto je stavba preventivně automatizována na potřebný počet obraných věží (Protoss Photon Canon) za uběhlý čas od počátku hry.

```
bool BuildUsefullBuildings()
{
    BWAPI::UnitType building;
    bool buildingWasSelected = false;
    bool buildCanon = false;
    bool buildingBuildResult = false;

    if (IsNexusNeeded() && (buildingWasSelected == false))
    {
        building = BWAPI::UnitTypes::getUnitType("Protoss␣Nexus");
    }

    if (IsPylonNeeded() && (buildingWasSelected == false))
    {
        building = BWAPI::UnitTYpes::getUnitType("Protoss␣Pylon");
        buildingWasSelected = true;
    }

    if (IsAssimilatorNeeded() && (buildingWasSelected == false))
    {
        building = BWAPI::UnitTypes::getUnitType("Protoss␣Assimilator");
        buildingWasSelected = true;
    }

    if (IsBaseDefenceNeeded() && (buildingWasSelected == false))
    {
        building = BWAPI::UnitTypes::getUnitType("Protoss␣Photon␣Canon");
        buildingWasSelected = true;
        buildCanon = true;
    }

    if (buildingWasSelected)
    {
        buildingBuildResult = BuildBuilding(building);
    }

    return true;
}
```

Výpis 3: Metoda **BuildUsefullBuildings()** mající na starost výrobu potřebných budov

5.2.3 Stavba plánovaných budov

Pro budovy, které nezávisí na aktuálním stavu rozvoje hráče byly StarCraft komunitou vytvořeny tzn. Build Ordery. Build Order je seznam budov a času ke každé budově, kdy by měly být vytvořeny. Build Ordery se mění v závislosti na způsobu hraní (agresivní, pasivní), doby na kterou jsou specializovány (early-game, mid-game nebo late-game) a na oponentovi(Zerg, Protoss či Terran) viz sekce **Strategie Rasy Protoss** 3.1. Pro stavbu

plánovaných budov slouží metoda **BuildBuildings()** viz. kód 4 . Tato metoda se nejprve podívá, jestli v poslední době nebyla zničena budova, která již stála, pokud ano, pak najde dostupného dělníka a přidelí mu nový rozkaz pro stavbu budovy. V případě, že žádná budova nebyla zničena, nebo všechny zničené budovy již byly postaveny se dá příkaz pro stavbu budovy, která má být aktuálně postavena(dle Build Orderu).

```
bool BuildBuildings()
    {
        std :: string  destroyedBuilding = m_map->m_buildings->GetDestroyedBuilding();
        std :: string  buildingToBuild = m_map->m_buildings->GetNextBuildingToBuild();
        int  buildingToBuildSupply = m_map->m_buildings->GetNextBuildingToBuildSupply();

        if (destroyedBuilding.size() > 6)
        {

            return BuildBuilding(BWAPI::UnitTypes::getUnitType(destroyedBuilding));
        }

        if (buildingToBuildSupply != 0)
        {

            return BuildBuilding(BWAPI::UnitTypes::getUnitType(buildingToBuild));
        }

        return true;
    }
```

Výpis 4: Metoda **BuildBuildings()** mající na starost výrobu budov neovlivněnou dynamickým stavem aktuální hry

5.3 Micromanagement

Micromanagement, neboli ovládání jednotek se skládá ze tří hlavních částí. První část se zabývá prozkoumáváním mapy. Činnost prozkoumávání je nutná zejména proto, jelikož prostředí RTS video her je pouze z části pozorovatelné (v daném čase je odhalena pouze ta část mapy, kde dohlédnou hráčovy jednotky) . Protihráč této nevědomosti může využít ve svůj prospěch a obsadit nedaleké doly na minerály, čímž se ekonomicky stane silnějším a umožní útok jednotek z více stran. Průzkum je také důležitý z hlediska informací o pozicích protihráčových jednotek a budov, jejichž počet a pozice napoví další nepřátelský tah a otevírá tímto nové možnosti útoku hráčových jednotek.

Útok samotný lze provést několika způsoby, kdy pro tuto práci byl zvolen herní strom, díky němuž je možné vytvořit určitou míru predikce budoucího stavu hry spolu s micromanagementem jednotlivých jednotek, jejichž útok je využíván k maximální efektivitě boje.

5.3.1 Průzkum

Pro průzkum (třída **Scout**) je z počátku hry zvolen dělník Protoss Probe, jelikož se jedná o nejlevnější a nejdostupnější jednotku za rasu Protoss s nadprůměrnou rychlostí pohybu.

Pro prozkoumávání byl vytvořen samostatný cílově orientovaný agent (implementace se nachází v souboru scout.h). Tento agent si vždy vyžádá dva workery, jednoho pro průzkum mapy a druhého pouze pro vyhledávání nepřítele. První průzkumník se používá v případě, že na poslední základně poblíž hlavní budovy Protoss Nexus docházejí minerály. Jakmile je jejich počet menší, než 6000 jednotek, pak se zavolá metoda ScoutingMinerals() a průzkumník se vydá na hledání. Veliká nevýhoda tohoto minerálového průzkumníka avšak spočívá v jeho pohybu, kdy prochází počáteční místa kde jsou vždy ukryty minerály, kdy průzkumník pokud není neviditelný(později Protoss Observer) utrpí velké ztráty a v některých případech je zabit.

Druhý průzkumník má za úkol nepřetržitě cestovat po hlavních částech mapy a hledat známky nepřátelské aktivity. Jedná se o cílově orientovaného agenta, který prochází místa na mapě a jakmile narazí na něpřátelskou jednotku, pak ji následuje. V případě, že jednotka může zaútočit na průzkumníka, si tento průzkumník udržuje bezpečnou vzdálenost.

Pokud je jakýkoliv z průzkumníků ztracen, pak se vyberou noví dělníci z aktuálně dostupných. V pozdějších fázích hry, kdy je možné postavit jednotku Protoss Observer, je tato jednotka postavena a nahrazuje práci průzkumné jednotky za dělníky.

5.3.2 Predikce boje

Pro předvídání výsledku následujícího střetu byl použit Herní strom(třída **GameTree**). Jak již bylo psáno výše, implementace herního stromu v RTS hrách je obtížná především z časového hlediska vytváření a procházení možností stromu. Narozdíl od šachů nebo dámy, kdy délka tahu je ovlivněna především stavem hry a zbývajícím časem do konce (vyhledávání ve stromu může trvat několik sekund) je herní strom neustále vytvářen(dynamický pohyb jednotek) a procházen, přičemž tyto činnosti nesmí zabírat více jak jednotky milisekund. Díky těmto důvodům byla na strom použita abstrakce v podobě shlukování jednotek v určitém dosahu do týmů při průměrné pozici a při maximálním počtu osmi týmů na každé straně (větší počet týmů vykazuje abnormální zpomalení FPS při hře a způsobuje její pády).

Při vytváření dílčího týmu jsou také vytvořeny proměnné reprezentující stav týmu (implementace ve třídě **AbstractedTeams**) jako součet životů všech jednotek, relativní pozice, typ jednotek, jejich zbroj a podobně.

Samotná funkce vytváření stromu by se dala popsat jako generování stromu o určité délce, kde každý uzel obsahuje hráčovy a nepřátelské abstraktní týmy spolu s předvídanými ztrátami na životech pro oba soupeře. Vytvoření nových uzlů mají na starost metody Attack a Flee, kdy obě metody nejprve vytvoří kopie všech abstraktních týmů a ztrát, poté provedou výpočty generující možnou budoucnost a následně všechny změny zaznamenají do nově vytvořeného uzlu (třída **State**).

Metody Attack() a Flee() ihned po duplikování abstraktních týmů hráče a protihráče provedou výpočty na jednotlivých týmech za určitý časový interval. Attack() vypočítá ztráty dané táhnoucí strany (hráč nebo nepřítel), kdy výpočet ztrát druhé strany se vypočítá časem potřebným k ujití vzdálenosti mezi jednotkami a následným počtem útoků, které abstraktní tým dokáže zasadit protivníkovi. Všechny změny pozic a ztrát na živo-

tech abstraktních týmů se při tomto zaznamenají do nově generovaného uzlu. Metoda Flee() má za úkol vypočítat ušlou vzdálenost zpět k základně za tentýž čas, co Attack().

Jelikož se pracuje s abstraktními třídami, tak výsledek nemusí být vždy přesný. Kupříkladu vypočítávání ušlé vzdálenosti nepoužívá žádný algoritmus z vyhledávání cest z důvodu časové náročnosti výpočtu. S tímto je tedy potřeba počítat.

Při prohledávání stromu byl použit algoritmus **Alpha Beta Prunning** ve třídě GameTree. Tento algoritmus prochází postupně uzel po uzlu, kdy klade důraz na to, jestli aktuální uzel označuje tah hráče nebo protivníka. Jeho výhoda spočívá v ořezávání částí stromu, do kterých ví, že se nepodívá.

5.3.3 Boj jednotek

Na základě výstupu predicke boje je jednotkám rozkázáno, zda-li útočit, nebo se stáhnout(implementace ve třídě **Micromanagement**. Původně byla zahrnuta ještě možnost strategický ústup na nejvohdnější místo, kde každá jednotka má za úkol najít nejbližší místo strategického významu (vyvýšená pozice, choke point). Avšak z důvodu časové náročnosti výpočtu Minmax stromu bylo od tohoto ustoupeno.

V případě útoku je každé aktuální dané bojové jednotce přiřazen nejbližší cíl na základě dosažitelnosti (nekteré pozemní jednotky nemohou útočit na létajícího nepřítele).

5.4 Výsledky robota

Z následujících výsledků je patrné, že implementovaná AI **mujRobot** je nejůčinější proti protivníkovi, který se orientuje na pozdější část hry.

Název robota	**Půměrné skóré**	**Typ hraní**	**Vítěz**
mujRobot	35200	Rush	WIN
Akilae Tribe	10980	Mid game	LOSE

Tabulka 2: mujRobot vs Akilae Tribe

Název robota	**Půměrné skóré**	**Typ hraní**	**Vítěz**
mujRobot	12500	Rush	LOSE
Velari Tribe	30500	Rush	WIN

Tabulka 3: mujRobot vs Velari Tribe

Název robota	**Půměrné skóré**	**Typ hraní**	**Vítěz**
mujRobot	14220	Rush	LOSE
Sargas Tribe	31570	Rush	WIN

Tabulka 4: mujRobot vs Sargas Tribe

6 Závěr

Hlavním cílem této bakalářské práce bylo navrhnout a naprogramovat AI robota do hry StarCraft: Broodwar a tím prozkoumat taje a zákoutí tohoto rozšiřujícího se fenoménu. AI robot byl naprogramován za pomoci několika známých metod používajících se pro vytvoření AI do her. Tohoto bylo úspěšně dosaženo. Robot samostatně plní svou funkci, staví základnu, kterou rozšiřuje, verbuje jednotky, prozkoumává oblast a pátra po nepříteli. Také bojová stránka robota byla splněna, kdy byla použita predikce říkající robotu, zda-li má útočit, nebo se stáhnout a vyrazit do boje při více jednotkách.

Jakožto jeho tvůrce na základě poznatků z tvorby již teď mohu usoudit, že vytvořit dokonalou AI i v tak uzavřeném prostředí jako je hra StarCraft je velice těžké nebo rovnou nemožné. Zjistil jsem, že použití některých algoritmů jako Alpha-Beta Prunning vede sice k nejlepšímu řešení ve hře šachy nebo dáma, avšak nikoliv v RTS hře jako StarCraft a to hlavně z důvodů mnoha nedefinovaných stavů které mohou nastat. Dále jsem zjistil, že tak jak platí pro týmy, že tým je silný tak jako jeho nejslabší hráč, stejně tak AI je silná tak, jako její nejslabší komponenta. Hra byla tedy rozdělena na komponenty zabývající se bojem a ekonomickou stránkou a bylo zjištěno, že pouhým nastavováním jednotlivých podmínek se dá dosáhnout různých výsledků například: odladěním počtu postavených obranných věží v závislosti na čase bylo docíleno k větším příjmům minerálů pro stavbu jednotek a tím získání rychlejší převahy na bojišti což vedlo k větší šanci na úspěch v samotné hře. Tyto původně velmi nepatrné věci tak mohou v kontečném důsledku znamenat vítězství nebo prohru nad nepřítelem. AI byla tvořena v BWAPI 3.7.4, které čas od času při testování shodilo celou hru (především z důvodu přístupu na špatnou adresu do paměti systému), nebo z nějasných důvodů všichni dělníci mající za úkol těžit surovinu zamrzli na místě a nic nedělali, což vedlo k zastavení dodávek surovin a tím ochromení celkové výroby bojových jednotek. Bohužel jeden z cílů : přihlásit robota do soutěže nebyl možný a to z důvodu uzavření přihlášek do soutěže v době, kdy bakalářská práce ještě ani nebyla přidelena. Tímto také nemohli být plně otestováni roboti, kteří se účastnili turnaje.

Pro testování robota byly tedy využita již implementovaná umělá inteligence ve hře. Na základě výsledků z bitev mohu konstatovat, že vytvořený robot plní svou funkci. Při testování má robot téměř 100 procentní šanci na výhrů vůči nepříteli, který preferuje rozrůstání základny. Na druhou stranu oproti protivníkům využívajícím styl hry rush má robot pramalou šanci. Tento neblahý výsledek je především dán faktem, že build order(pořadí stavby budov) robota je tvořen na základě sledování zápasů mezi lidmi, kterým trvá o poznání delší dobu, než započnou s útokem.

Do budoucna plánuji robota vylepšit, předělat některé algoritmy, které se při tvorbě neosvědčily a následně jej přihlásit do turnaje Umělé Inteligence ve StarCraft: Broodwar.

7 Reference

[1] Goossens, Michel, *The LaTeX companion*, New York: Addison, 1994.

[2] Lamport, Leslie, *LaTeX: a document preparation system: user's guide and reference manual*, New York: Addison-Wesley Pub. Co., 1994.

[3] CHUNG, Michael - BURO, Michael - SCHAEFFER, Jonathan *Monte Carlo Planning in RTS Games*, Department of Computing Science, University of Alberta, Admonton, Alberta, Canada

[4] RUSSEL, Stuart. - NORVIG, Peter, *Artificial Inteligence - A Modern Approach, Third Edition*, New Jersey: Prentice Hall, 2009, ISBN 0-13-604259-7, Kapitola.

[5] RUSSEL, Stuart. - NORVIG, Peter, *Artificial Inteligence - A Modern Approach, Third Edition*, New Jersey: Prentice Hall, 2009, ISBN 0-13-604259-7, Kapitola.

[6] ROBERTSON, Glen. - WATSON, Ian, *A Review of Real-Time Strategy Game AI*, New Zealand: University of Auckland.

[7] *David, POOLE - Alan, Mackworth. Artificial Intelligence - foundations of computational agents*[online], 2010 <www.artint.info/html/ArtInt_262.html>

[8] *Ben G., WEBER - Michael, MATEAS - Arnav, JHALA. Applying Goal-Driven Autonomy to StarCraft*[online], <www.aaai.org/ocs/index.php/AIIDE/AIIDE10/paper/2142

[9] *Game Guide - StarCraft II*[online], <eu.battle.net/sc2/en/game/>

[10] *BNL | History"The First Video Game?*[online], <www.bnl.gov/about/history/firstvideo.php>

[11] *David, WINTER. Pong-Story: Atari PONG - First steps*[online], 2013 <www.bnl.gov/about/history/firstvideo.php>

[12] *bwapi: An API for interacting with Starcraft: Broodwar (1.16.1)*[online], <https://code.google.com/p/bwapi/>

A Přílohy

Příloha na CD, popis v sekci 5.1.1.